WEB TECHNOLOGY AND ITS APPLICATIONS

MODULE 7 – Ajax-SYLABUS

- Introduction
- advantages & disadvantages
- Purpose of it
- ajax based web application
- alternatives of ajax

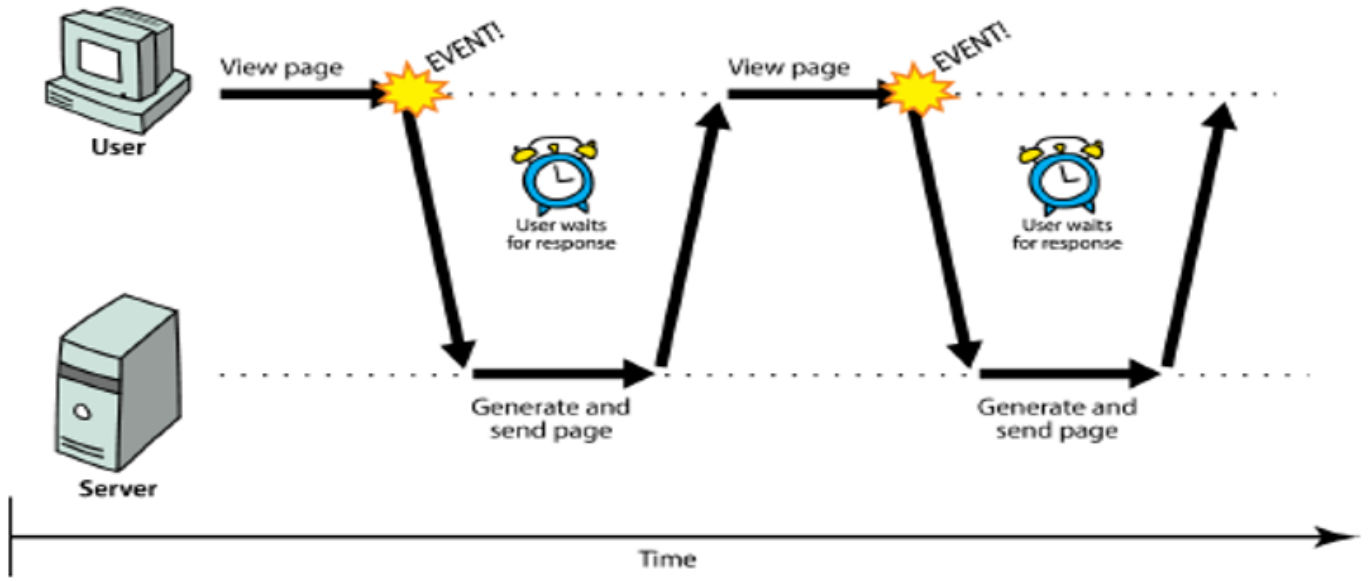
JavaScript in Modern Times- Introduction AJAX

JavaScript became a much more important part of web development in the mid 2000s with **AJAX**.

AJAX is both an acronym as well as a general term.

- As an acronym it means **A**synchronous **J**avaScript **A**nd **X**ML.
- The most important feature of AJAX sites is the asynchronous data requests.

Synchronous web communication



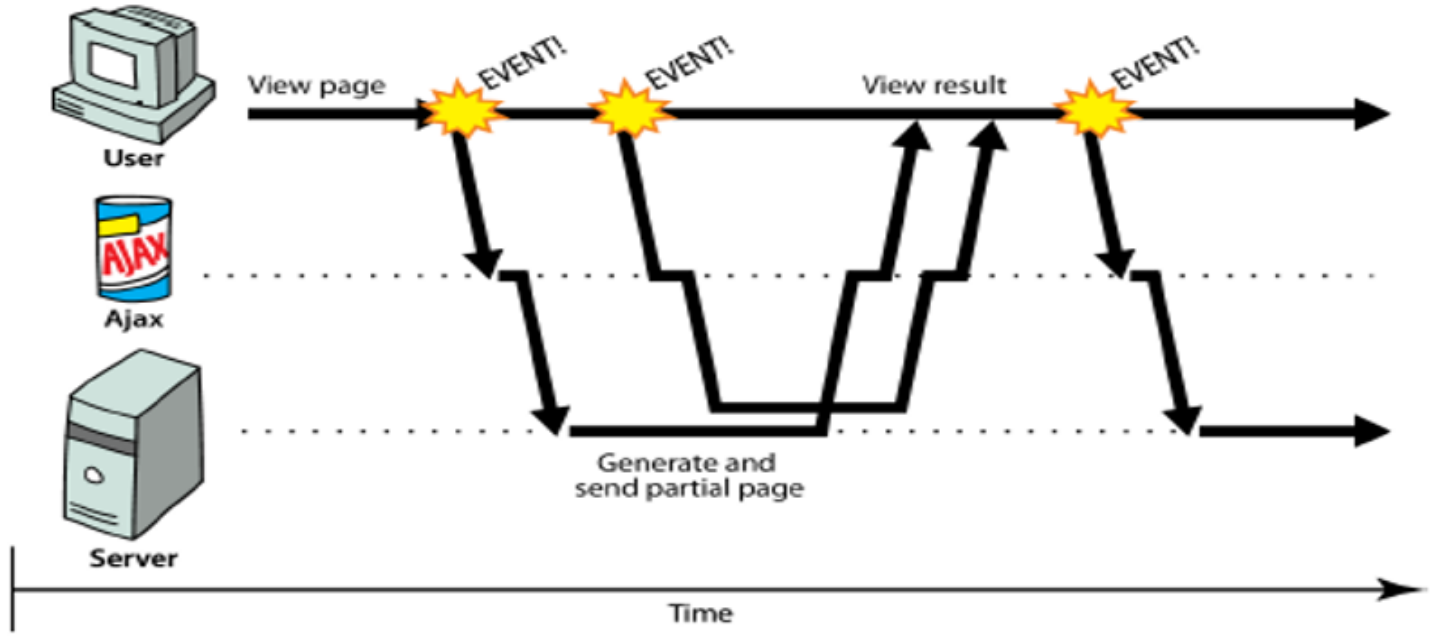
- synchronous: user must wait while new pages load
  - the typical communication pattern used in web pages (click, wait, refresh)

Purpose of AJAX

Web applications and Ajax

- **Ajax:** Asynchronous JavaScript and XML
  - not a programming language; a particular way of using JavaScript
  - downloads data from a server in the background
  - allows dynamically updating a page without making the user wait
  - avoids the "click-wait-refresh" pattern
  - Example: Google Suggest

Asynchronous web communication



- **asynchronous:** user can keep interacting with page while data loads
  - communication pattern made possible by Ajax

AJAX Technologies

HTML
- Used to build web forms and identify fields

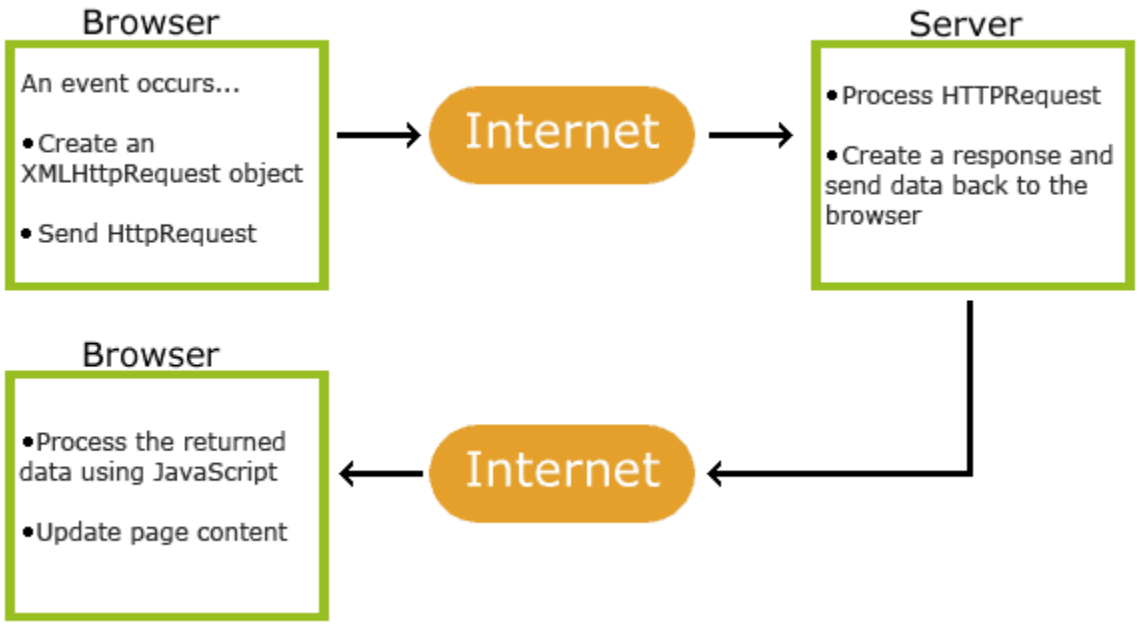
Javascript
- Facilitates asynchronous communication and modification of HTML in-place

DHTML - Dynamic HTML
- Additional markup for modifying and updating HTML

DOM - Document Object Model
- Used via Javascript to work with both the structure of your HTML and also XML from the server

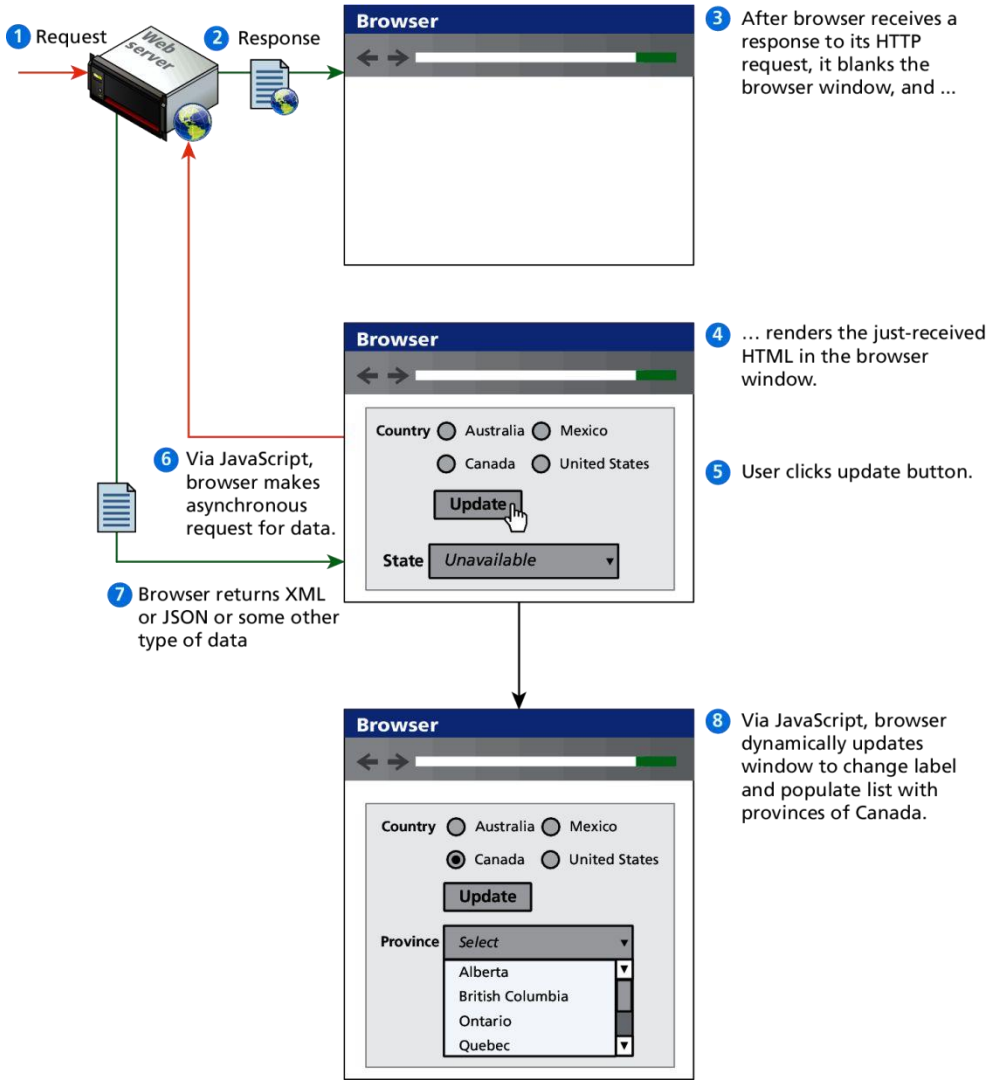
How AJAX Works:

- User Interaction:
  - A user interacts with a web page (e.g., clicks a button).
- AJAX Request:
  - JavaScript makes an HTTP request to the server (using XMLHttpRequest or the newer fetch API).
- Server Response:
  - The server processes the request and sends back data (often in JSON format).
- Update Web Page:
  - JavaScript processes the server's response and updates the web page without reloading.


Browser
An event occurs...
• Create an XMLHttpRequest object
• Send HttpRequest
Internet
Server
• Process HTTPRequest
• Create a response and send data back to the browser
Browser
• Process the returned data using JavaScript
• Update page content
Internet

Asynchronous data requests

The better AJAX way



Ajax based web application

What is AJAX?

AJAX = **A**synchronous **J**avaScript **A**nd **X**ML.
AJAX is not a programming language.

AJAX just uses a combination of:
•A browser built-in XMLHttpRequest object (to request data from a web server)
•JavaScript and HTML DOM (to display or use the data)

The XMLHttpRequest Object

Base object for AJAX

- Used to make connections, send data, receive data, etc.

Allows your javascript code to talk back and forth with the server all it wants to, without the user really knowing what is going on.

Available in most browsers

- But called different things

Create an XMLHttpRequest Object

Syntax for creating an XMLHttpRequest object:

```
variable = new XMLHttpRequest();
```

Example

```
var xhttp = new XMLHttpRequest();
```

XMLHttpRequest Object Methods

Method	Description
new XMLHttpRequest()	Creates a new XMLHttpRequest object
abort()	Cancels the current request
getAllResponseHeaders()	Returns header information
getResponseHeader()	Returns specific header information
open(method,url,async,user,psw)	Specifies the request method: the request type GET or POST url: the file location async: true (asynchronous) or false (synchronous) user: optional user name psw: optional password
send()	Sends the request to the server Used for GET requests
send(string)	Sends the request to the server. Used for POST requests
setRequestHeader()	Adds a label/value pair to the header to be sent

XMLHttpRequest Object Properties

Property	Description
onreadystatechange	Defines a function to be called when the readyState property changes
readyState	Holds the status of the XMLHttpRequest. 0: request not initialized 1: server connection established 2: request received 3: processing request 4: request finished and response is ready
responseText	Returns the response data as a string
responseXML	Returns the response data as XML data
status	Returns the status-number of a request 200: "OK" 403: "Forbidden" 404: "Not Found" For a complete list go to the Http Messages Reference
statusText	Returns the status-text (e.g. "OK" or "Not Found")

Send a Request To a Server

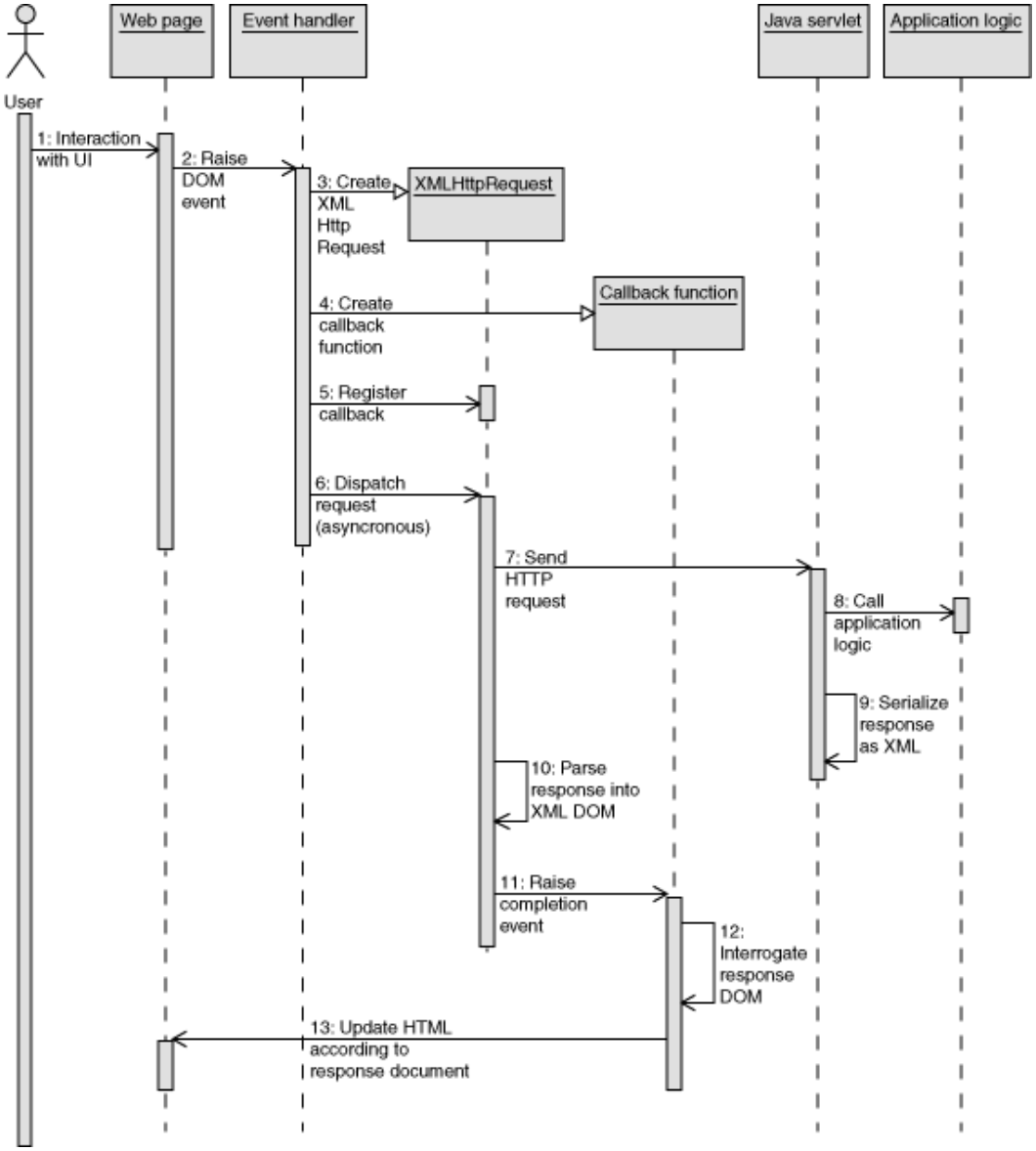
To send a request to a server, we use the open() and send() methods of the XMLHttpRequest object:

```
xhttp.open("GET", "ajax_info.txt", true);
xhttp.send();
```

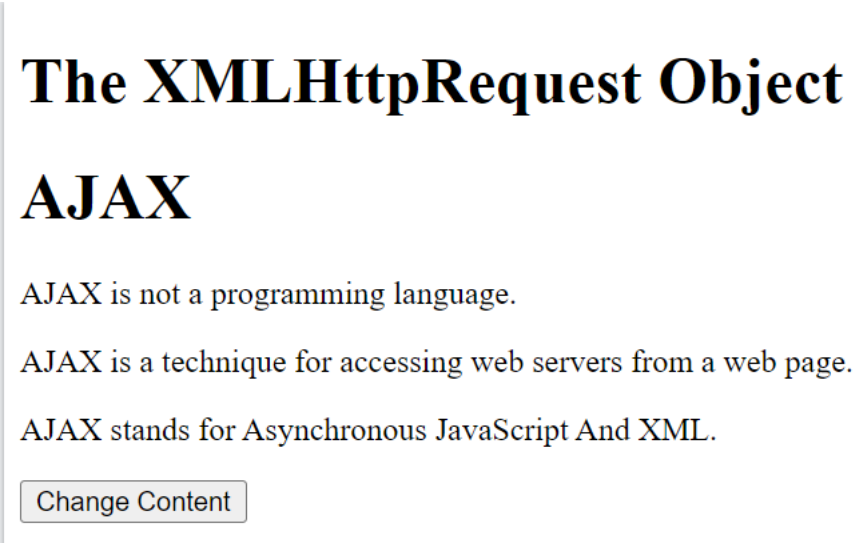
Method	Description
open(*method, url, async*)	Specifies the type of request *method*: the type of request: GET or POST *url*: the server (file) location *async*: true (asynchronous) or false (synchronous)
send()	Sends the request to the server (used for GET)
send(*string*)	Sends the request to the server (used for POST)

Typical AJAX Flow

- Make the call
  - Gather information (possibly from HTML form)
  - Set up the URL
  - Open the connection
  - Set a callback method
  - Send the request
- Handle the response (in callback method)
  - When request. readyState == 4 and request.status == 200
  - Get the response in either text or xml
    - request.responseText or request.responseXML
  - Process the response appropriately for viewing
  - Get the objects on the page that will change
    - document.getElementById or document.getElementByName, etc.
  - Make the changes


User
Web page
Event handler
XMLHttpRequest
Callback function
Java servlet
Application logic
1: Interaction with UI
2: Raise DOM event
3: Create XML Http Request
4: Create callback function
5: Register callback
6: Dispatch request (asyncronous)
7: Send HTTP request
8: Call application logic
9: Serialize response as XML
10: Parse response into XML DOM
11: Raise completion event
12: Interrogate response DOM
13: Update HTML according to response document

```
<!DOCTYPE html>
<html>
<body>
<div id="demo">
<h1>The XMLHttpRequest Object</h1>
<button type="button" onclick="loadDoc()">Change Content</button>
</div>
<script>
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200) {
      document.getElementById("demo").innerHTML =
      this.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
</script>
</body>
</html>
```



HTML Page

```
<!DOCTYPE html>
<html>
<body>

<div id="demo">
  <h2>Let AJAX change this text</h2>
  <button type="button" onclick="loadDoc()">Change
Content</button>
</div>

</body>
</html>
```

The function requests data from a web server and displays it:

```
function loadDoc() {
  var xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function() {
    if (this.readyState == 4 && this.status == 200)
{
     document.getElementById("demo").innerHTML = th
is.responseText;
    }
  };
  xhttp.open("GET", "ajax_info.txt", true);
  xhttp.send();
}
```

The "ajax_info.txt" file used in the example above, is a simple text file and looks like this:

```
<h1>AJAX</h1>
<p>AJAX is not a programming language.</p>
<p>AJAX is a technique for accessing web servers from a web page.</p>
<p>AJAX stands for Asynchronous JavaScript And XML.</p>
```

AJAX has both advantages and disadvantages:

Advantages :

1. Improved user experience: AJAX can provide a smoother experience by updating parts of a page without reloading the entire page.
2. Reduced bandwidth usage: AJAX reduces the amount of traffic exchanged between the client and the web server because it only exchanges the HTML that's relevant to the section of the page being updated.
3. Faster response times: AJAX enables faster response times by allowing only the necessary information to be transmitted to the server.
4. Asynchronous processing: AJAX allows web applications to receive and send data from the web server simultaneously.

AJAX has both advantages and disadvantages:

Disadvantages :

1. Increased complexity: AJAX can be more complex to implement.
2. Security vulnerabilities:AJAX can introduce potential security vulnerabilities.
3. Browser compatibility:AJAX may not be accessible to users of older browsers or screen readers.
4. Search engine optimization: AJAX applications may not be indexed by search engines.

Alternatives to AJAX

1. Fetch API: Streamlining Data Retrieval:

The Fetch API is a modern JavaScript interface that simplifies making network requests. It provides a more straightforward and promise-based approach compared to AJAX. With its native support for Promises, it offers improved error handling and cleaner code structure.

2. WebSockets: Real-time Communication:

WebSockets provide bidirectional, real-time communication between the client and server. This technology is ideal for applications that require instant updates, such as online gaming, chat applications, or collaborative tools.

Alternatives to AJAX

3.GraphQL: Efficient Data Fetching:

GraphQL is a query language for APIs that allows clients to request precisely the data they need. Unlike traditional REST APIs, which often over-fetch data, GraphQL enables efficient data fetching, reducing the payload and improving performance.

4. Server-Sent Events (SSE):

SSE is a technology that enables a server to send real-time updates to a client using a single HTTP connection.

5. WebRTC: Peer-to-Peer Communication:

WebRTC empowers web applications with peer-to-peer communication capabilities. It’s ideal for building video conferencing, file sharing, and other real-time collaboration tools directly in the browser.